This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

IN THE CLAIMS:

The following listing of claims will replace all prior versions, and listings, of claims in the application.

1. (Currently Amended) A method for transferring data over an external transmission medium, the method comprising:

receiving a plurality of transfer requests;

building a chain of transfer objects, wherein each transfer object corresponds to one of the plurality of transfer requests, wherein said building the chain of transfer objects includes linking the plurality of transfer objects together sequentially; and

sequentially performing the request of each transform object in the chain of transfer objects ~~sequentially~~.

2. (Currently Amended) The method of claim 1, wherein the building the chain of transfer objects comprises:

creating a plurality of transfer objects; and

attaching a user callback function to zero or more of the plurality of transfer objects~~; and~~

~~linking the plurality of transfer objects together sequentially~~.

3. (Original) The method of claim 2, wherein said creating the plurality of transfer objects comprises:

allocating memory for each of the plurality of transfer objects;

populating each of the plurality of transfer objects with transfer information, wherein the transfer information comprises one or more transfer types;

allocating memory for a plurality of request block objects, wherein each of the plurality of request block objects corresponds to one of the plurality of transfer objects, wherein each of the request block objects is comprised in a corresponding one of the transfer objects, and wherein each of the request block objects provides an operating

system-independent and bus-independent interface which encapsulates operating system-dependent and bus-dependent data; and

populating each of the plurality of request blocks with operating system-dependent and bus-dependent information related to a request of a corresponding transfer object.

4. (Original) The method of claim 3, wherein the one or more transfer types comprise one or more of a read transfer, a write transfer, a single point transfer, a block transfer, a synchronous transfer, an asynchronous transfer, a random read transfer, or a random write transfer.

5. (Currently Amended) The method of claim 1, wherein [[the]] said sequentially performing the request of each transform object in the chain of transfer objects ~~sequentially~~ reduces passive/dispatch level transitions.

6. (Currently Amended) The method of claim 1, wherein [[the]] said sequentially performing the request of each transform object in the chain of transfer objects ~~sequentially~~ reduces user/kernel mode transitions.

7. (Currently Amended) The method of claim 1, wherein [[the]] said sequentially performing the request of each transform object in the chain of transfer objects ~~sequentially~~ comprises performing a request of a current transfer object;

wherein said performing the request of the current transfer object is executed on a first thread at passive level if the current transfer object is the first transfer object in the chain; and

wherein said performing the request of the current transfer object is executed on a system thread at kernel-dispatch level if the current transfer object is not the first transfer object in the chain.

8. (Currently Amended) The method of claim 7, wherein [[the]] said sequentially performing the request of the current transform object further comprises:

executing a static callback function to return control to the current transfer object on the system thread at kernel dispatch level;

executing a first callback function of the current transfer object to complete the request on the system thread at kernel dispatch level;

executing the first callback function of the current transfer object to set an event on the system thread at kernel dispatch level, wherein the event signals completion of the request;

determining if there is a user callback function attached to the current transfer object; and

executing the user callback function on the system thread at kernel dispatch level if there is a user callback function attached to the current transfer object.

9. (Original) The method of claim 8,

wherein said performing the request of the first transfer object on the first thread at passive level comprises performing the request of the first transfer object on the first thread at kernel-passive level.

10. (Original) The method of claim 8,

wherein said performing the request of the first transfer object on the first thread at passive level comprises performing the request of the first transfer object on the first thread at user-passive level.

11. (Currently Amended) The method of claim 8, further comprising:

a last transfer object in the chain calling a wait function on the first thread after the performing the request of the first transfer object in the chain of transfer objects, wherein the calling the wait function on the first thread puts the first thread into a sleep mode;

wherein the first thread is awakened from the sleep mode in response to [[d)]] the current transfer object setting an event on the system thread at kernel dispatch level, wherein the event signals completion of the request; and

wherein the current transfer object is the last transfer object in the chain;

12. (Original) The method of claim 1,

wherein said external transmission medium comprises an IEEE 1394 bus, wherein said IEEE 1394 bus is implemented in accordance with an IEEE 1394 protocol specification.

13. (Original) The method of claim 1,

wherein said external transmission medium comprises a Universal Serial Bus (USB) bus.

14. (Original) The method of claim 1,

wherein said external transmission medium uses the Ethernet protocol.

15. (Original) A system for transferring data, the system comprising:

a host system including a processor and memory, wherein the memory stores data and driver software, and wherein the processor is operable to execute the driver software;

an external communications medium; and

a device, wherein the device is coupled to the host system through the external transmission medium;

wherein the driver software is executable to receive a plurality of transfer requests, build a chain of transfer objects, and submit the chain of transfer objects for execution by the host system;

wherein each of the plurality of transfer objects corresponds to one of the plurality of transfer requests, and wherein each of the transfer objects is operable to perform a request to read from or write to the device; and

wherein the host system is operable to perform the request of each transfer object in the chain of transfer objects sequentially, wherein the request of the first transfer object in the chain of transfer objects is executed on a thread at passive level, and wherein the requests of subsequent transfer objects in the chain of transfer objects are executed on one or more system threads at kernel-dispatch level.

16. (Original) The system of claim 15,

wherein each of the plurality of transfer objects comprises:

transfer information, wherein the transfer information includes one or more transfer types;

a request block object, wherein the request block object provides an operating system-independent and bus-independent interface which encapsulates operating system-dependent and bus-dependent data related to a request of a corresponding transfer object, and wherein the request block object implements the external communications medium protocol in an operating system specific data structure;

a link which is operable to provide access to another transfer object;

an intrinsic callback function; and

a static callback function;

17. (Original) The system of claim 16,

wherein one or more of the plurality of transfer objects each further comprises a user callback function.

18. (Original) The system of claim 15,

wherein the software being executable to build a chain of transfer objects includes the software being executable to:

allocate memory for each of the plurality of transfer objects;

populate each of the plurality of transfer objects with transfer information which comprises one or more transfer types;

allocate memory for a plurality of request blocks, wherein each of the plurality of request blocks corresponds to one of the plurality of transfer objects, and wherein each of the request blocks is comprised in a corresponding one of the transfer objects, wherein each of the request block objects provides an operating system- and bus-independent interface which encapsulates operating system- and bus-dependent data;

populate each of the plurality of request blocks with operating system- and bus-dependent information relating to the request of the transfer object;

attach a user callback function to zero or more of the plurality of transfer objects; and

chain the plurality of transfer objects together sequentially.

19. (Original) The system of claim 15,

wherein the host system being operable to perform the request of each transform object in the chain of transfer objects sequentially comprises the host system being operable to perform a request of a current transfer object on a first thread at passive level if the current transfer object is the first transfer object in the chain; and perform the request of the current transfer object on a system thread at kernel-dispatch level if the current transfer object is not the first transfer object in the chain.

20. (Original) The system of claim 19, wherein the host system being operable to perform the request of the current transfer object further comprises the host system being operable to:

execute a static callback function to return control to the current transfer object on the system thread at kernel dispatch level;

execute the intrinsic callback function of the current transfer object to complete the request on the system thread at kernel dispatch level;

execute the intrinsic callback function of the current transfer object to set an event on the system thread at kernel dispatch level, wherein the event signals completion of the request;

determine if there is a user callback function attached to the current transfer object;

execute the user callback function of the current transfer object on the system thread at kernel dispatch level if there is a user callback function attached to the current transfer object; and

21. (Original) The system of claim 19,

wherein the host system being operable to perform the request of the first transfer object on the first thread at passive level comprises the host system being operable to perform the request of the first transfer object on the first thread at kernel-passive level.

22. (Original) The system of claim 19,

wherein the host system being operable to perform the request of the first transfer object on the first thread at passive level comprises the host system being operable to perform the request of the first transfer object on the first thread at user-passive level.

23. (Original) The system of claim 19,

wherein the host system is further operable to execute a wait function of a last transfer object in the chain on the first thread after the performing the request of the first transfer object in the chain of transfer objects, wherein the executing the wait function of the last transfer object in the chain on the first thread puts the first thread into a sleep mode; and

wherein the first thread is awakened from the sleep mode in response to d) executing the intrinsic callback function of the current transfer object to set an event on the system thread at kernel dispatch level, wherein the event signals completion of the request, and wherein the current transfer object is the last transfer object in the chain.

24. (Original) The system of claim 15,

wherein the host system comprises a computer system.

25. (Original) The system of claim 15, wherein the one or more transfer types comprise one or more of a read transfer, a write transfer, a single point transfer, a block transfer, a synchronous transfer, an asynchronous transfer, a random read transfer, or a random write transfer.

26. (Original) The system of claim 15,

wherein said external transmission medium comprises an IEEE 1394 bus, and wherein said IEEE 1394 bus is implemented in accordance with an IEEE 1394 protocol specification.

27. (Original) The system of claim 15,

wherein said external transmission medium comprises a Universal Serial Bus (USB) bus.

28. (Original) The system of claim 15,

wherein said external transmission medium uses the Ethernet protocol.

29. (Original) The system of claim 15,

wherein the driver software comprises data acquisition driver software, and wherein the device comprises a data acquisition device.

30. (Original) The system of claim 15, further comprising a toolbox stored in the memory of the host computer, wherein the toolbox is operable to provide a generic interface to manage communications between drivers.

31. (Original) A transfer object, wherein the transfer object is configurable to encapsulate data transfer-related functionality to provide a generic interface for transmission of data over a variety of external transmission media and protocols, comprising:

transfer information, wherein the transfer information includes one or more transfer types;

a request block object, wherein the request block object provides an operating system-independent and bus-independent interface which encapsulates operating system-dependent and bus-dependent data related to a data transfer request, and wherein the request block object implements the external transmission medium protocol in an operating system specific data structure;

a link which is operable to provide access to another transfer object, wherein the link provides a mechanism for chaining transfer objects together;

an intrinsic callback function which is operable to execute a transaction to complete the data transfer request, and which is further operable to set an event signaling completion of the data transfer request; and

a static callback function;

32. (Original) The transfer object of claim 31, further comprising:

a user callback function, wherein the user callback function is operable to execute a user-specified task after the data transfer request of the transfer object has been executed.

33. (Original) The transfer object of claim 31, wherein the one or more transfer types comprise one or more of a read transfer, a write transfer, a single point transfer, a block transfer, a synchronous transfer, an asynchronous transfer, a random read transfer, or a random write transfer.